EE219 Project 1

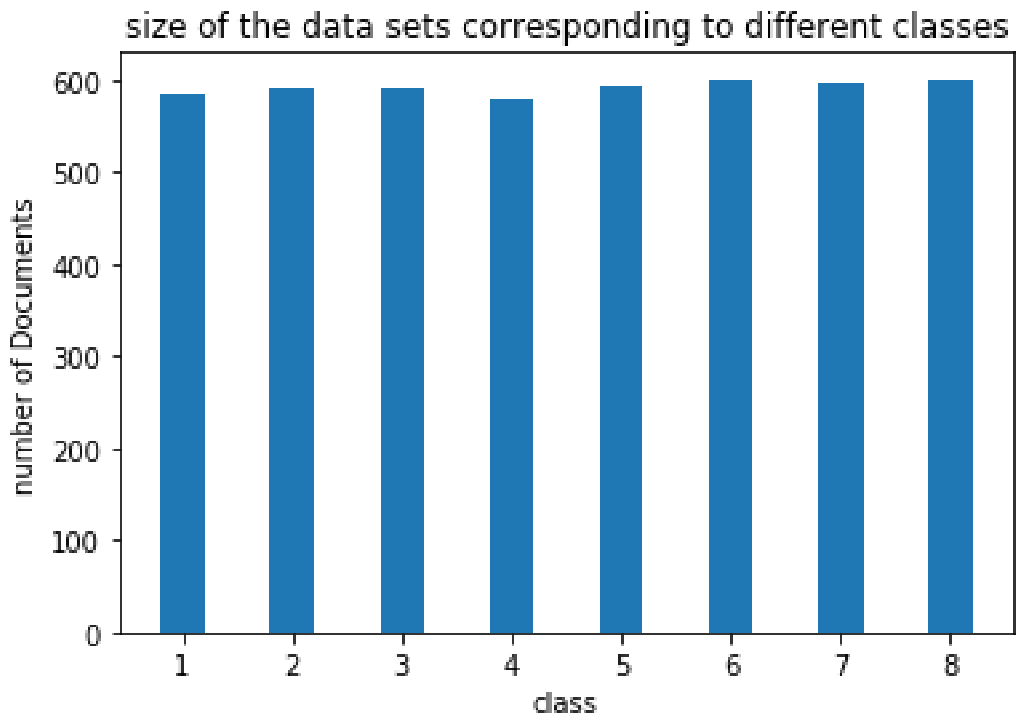
Classification Analysis on Textual Data
Winter 2018

Xingyi Chen 205032924
Wenfei Lu 505035450

problem a:

We get documents of 8 classes and then plot a histogram of the number of training documents per class.

computer technology	recreational activity
1.comp.graphics	5.rec.autos
2.comp.os.ms-windows.misc	6.rec.motorcycles
3.comp.sys.ibm.pc.hardware	7.rec.sport.baseball
4.comp.sys.mac.hardware	8.rec.sport.hockey



As it shows, the documents in 8 classes are evenly distributed so we do not need to handle the imbalance issue of data.

problem b:

In this part, we generate a TFxIDF matrix. First, we define an analyzer with the function of removing characters other than letters, stemming words, and removing stop words and punctuations. Then we use CountVectorizer() and TfidfTransformer() to generate a TFxIDF matrix with setting min_df to 2 and 5.

When min_df=2, the shape of the TFxIDF matrix is (4154, 12385), which means that the document number is 4145 and the term number is 12385.

When min_df=5, the shape of the TFxIDF matrix is (4154, 5545), which means that the term number is 5545.

Since the min_df means that terms with term number which is lower than the given threshold should not be included, it’s easy to understand why the term number of TFxIDF matrix with min_df=5 is much smaller than why the term number of TFxIDF matrix with min_df=2.

problem c:

To find the top 10 significant terms for the given classes, we first generate a TFxICF matrix and then extract the top 10 significant terms. In the first part, ,since we only care about the terms with big term frequency, it’s safe to set min_df=5. The results are sorted and stored as below:

comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	misc.forsale	soc.religion.christian
'scsi'	'mac'	'wolverin'	'god'
'drive'	'use'	'new'	'christian'
'use'	'appl'	'sale'	'church'
'ide'	'drive'	'offer'	'jesus'
'mb'	'scsi'	'use'	'christ'
'card'	'problem'	'dos'	'sin'
'disk'	'mhz'	'ship'	'homosexu'
'control'	'monitor'	'includ'	'peopl'
'dos'	'mb'	'price'	'say'
'dx'	'quadra'	'drive'	'believ'

problem d:

By using LSI and NMF, we reduce the dimensionality of TFxIDF matrix to a 50-dimensional vec tor. The output shape of the matrix is (4154, 50).

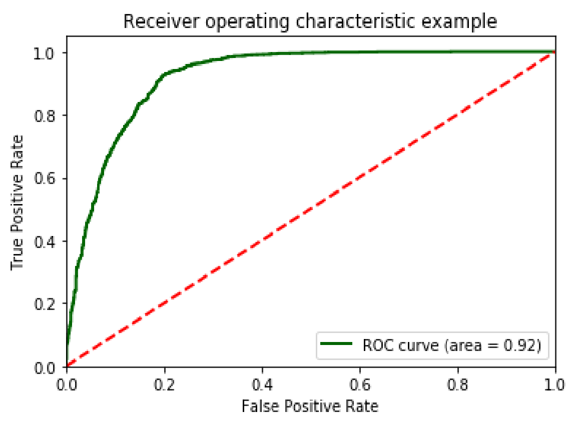
For problem e-i (except g), we compare LSI with NMF based on min_df=5 and min_df=2.

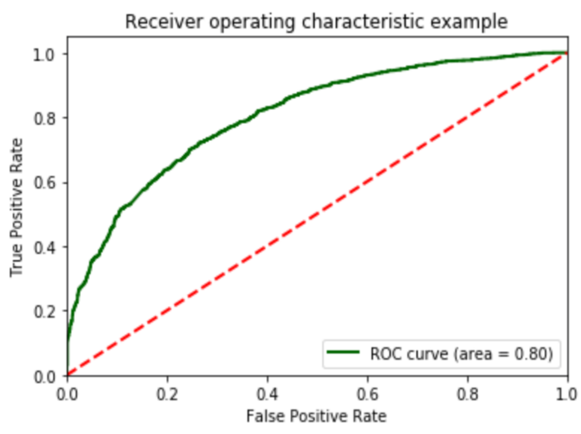
For problrm g, we use NMF with min_df=2.

Firstly, we compare LSI with NMF with min_df=5. Figures on the left side are based on LSI while the right side are NMF.

problem e:

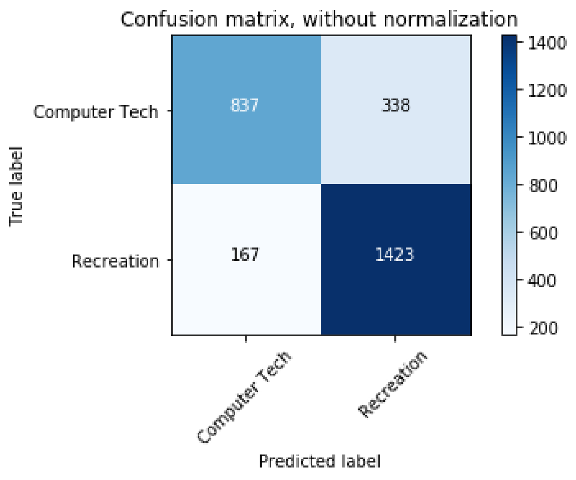
Hard margin SVC λ = 1000

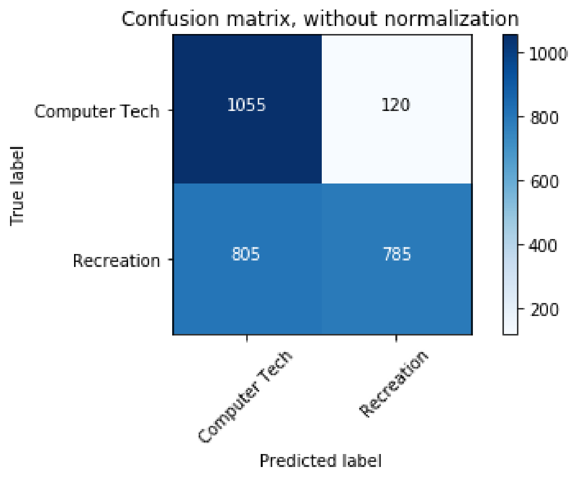




```
Confusion matrix, without normalization
[[ 837  338]
 [ 167 1423]]
Normalized confusion matrix
[[0.71 0.29]
 [0.11 0.89]]
```

```
Confusion matrix, without normalization
[[1055  120]
 [ 805  785]]
Normalized confusion matrix
[[0.9  0.1 ]
 [0.51 0.49]]
```

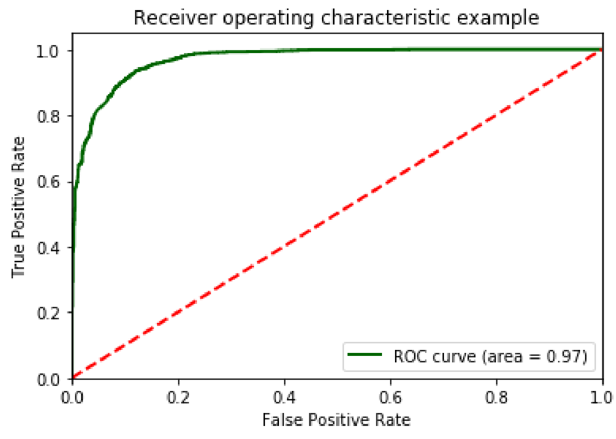


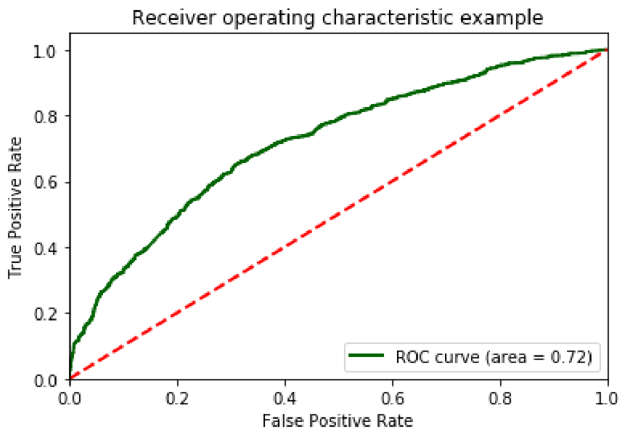


```
The accuracy score is: 0.8173598553345389
The recall score is: 0.8949685534591195
The precision score is: 0.8080636002271436
```

```
The accuracy score is: 0.6654611211573237
The recall score is: 0.4937106918238994
The precision score is: 0.8674033149171271
```

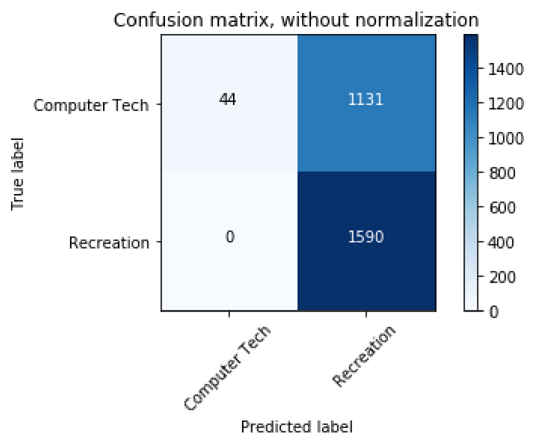
Soft margin SVC: λ = 0.001

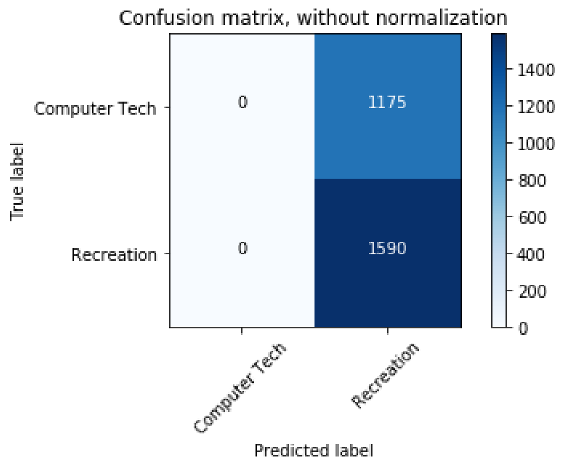




```
Confusion matrix, without normalization
[[  44 1131]
 [   0 1590]]
Normalized confusion matrix
[[0.04 0.96]
 [0.   1.  ]]
```

```
Confusion matrix, without normalization
[[    0 1175]
 [    0 1590]]
Normalized confusion matrix
[[0. 1.]
 [0. 1.]]
```





```
The accuracy score is: 0.5909584086799277
The recall score is: 1.0
The precision score is: 0.5843439911797134
```

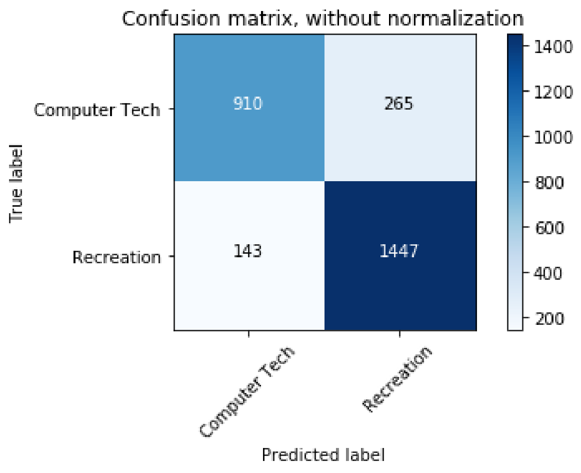
```
The accuracy score is: 0.5750452079566004
The recall score is: 1.0
The precision score is: 0.5750452079566004
```

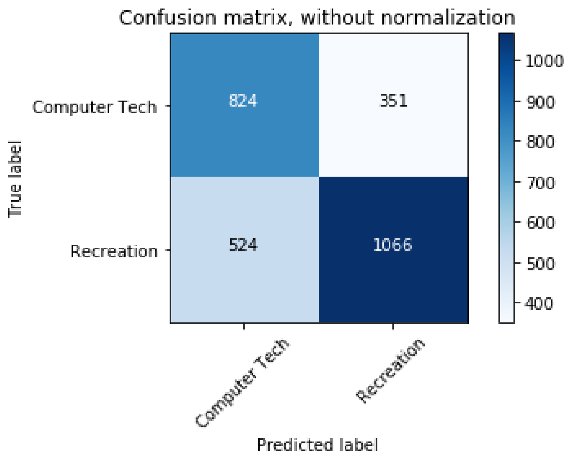
problem f:

We divide the data with 5-fold. For k =-3, -2, -1, 0, 1, 2, 3, we can get the mean score 0.58, 0.86, 0.93, 0.94, 0.95, 0.94, 0.94 separately. **The best value of parameter λ is 10.**

```
Confusion matrix, without normalization
[[ 910  265]
 [ 143 1447]]
Normalized confusion matrix
[[0.77 0.23]
 [0.09 0.91]]
```

```
Confusion matrix, without normalization
[[ 824  351]
 [ 524 1066]]
Normalized confusion matrix
[[0.7  0.3 ]
 [0.33 0.67]]
```





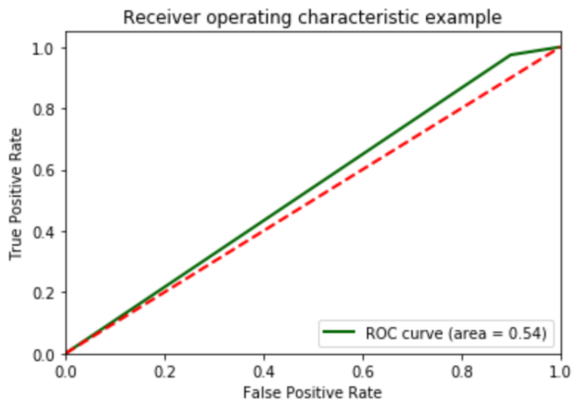
```
The accuracy score is: 0.5909584086799277
The recall score is: 1.0
The precision score is: 0.5843439911797134
```

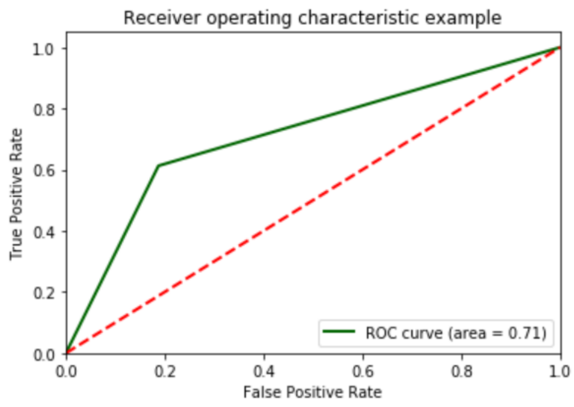
```
The accuracy score is: 0.5750452079566004
The recall score is: 1.0
The precision score is: 0.5750452079566004
```

problem g:

In this part, we use 2 different Naïve Bayes functions (Multinomial Naïve Bayes and Gaussian Naïve Bayes) to see which one is better. We implemented NMF since Multinomial Naïve Bayes only allows non-negative matrix.

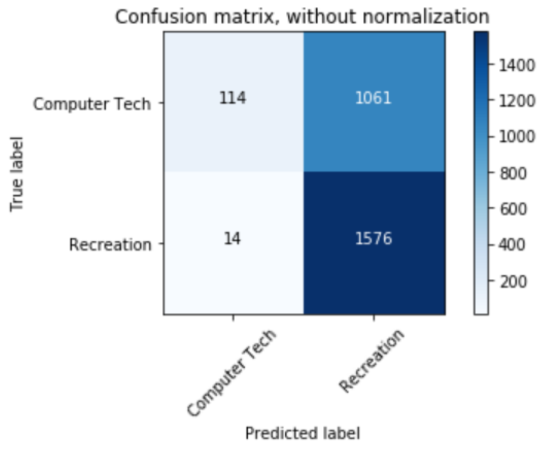
The result are as follows. From the result, we can see that Gaussian Naïve Bayes is better since the accuracy is higher than that of Multinomial Naïve Bayes.

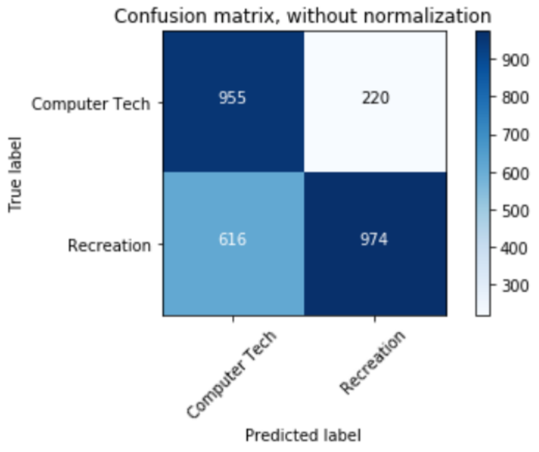




```
Confusion matrix, without normalization
[[ 114 1061]
 [  14 1576]]
Normalized confusion matrix
[[0.1  0.9 ]
 [0.01 0.99]]
```

```
Confusion matrix, without normalization
[[955 220]
 [616 974]]
Normalized confusion matrix
[[0.81 0.19]
 [0.39 0.61]]
```





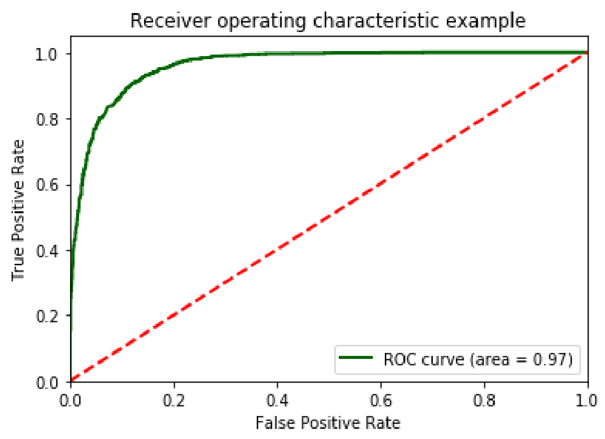
```
The accuracy score is: 0.6112115732368897
The recall score is: 0.9911949685534591
The precision score is: 0.5976488433826318
```

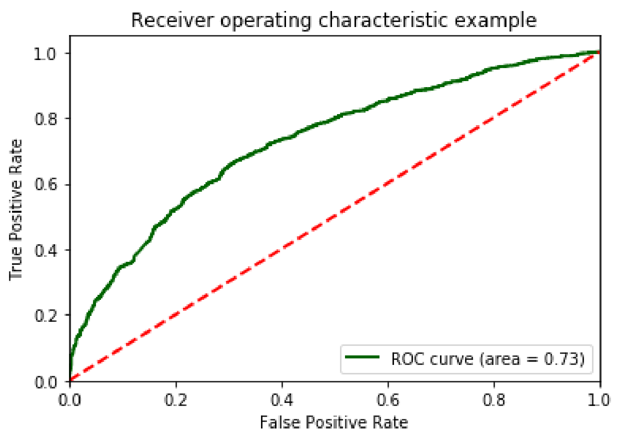
```
The accuracy score is: 0.6976491862567812
The recall score is: 0.6125786163522012
The precision score is: 0.8157453936348409
```

problem h:

For logistic regression classifier:
With the default norm regularization l2, we get the ROC curve as following.

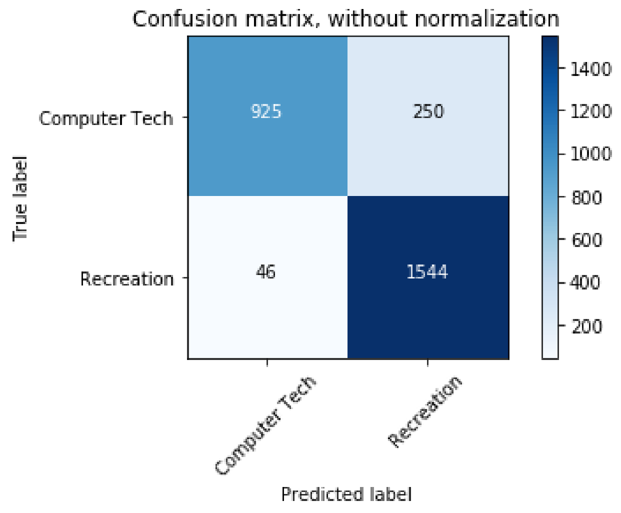
ROC curve:

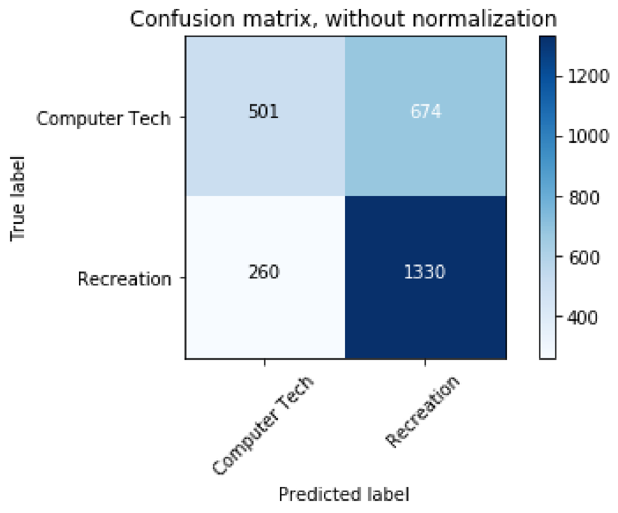




```
Confusion matrix, without normalization
[[ 925  250]
 [  46 1544]]
Normalized confusion matrix
[[0.79 0.21]
 [0.03 0.97]]
```

```
Confusion matrix, without normalization
[[ 501  674]
 [ 260 1330]]
Normalized confusion matrix
[[0.43 0.57]
 [0.16 0.84]]
```





```
The accuracy score is: 0.8929475587703436
The recall score is: 0.9710691823899371
The precision score is: 0.8606465997770345
```

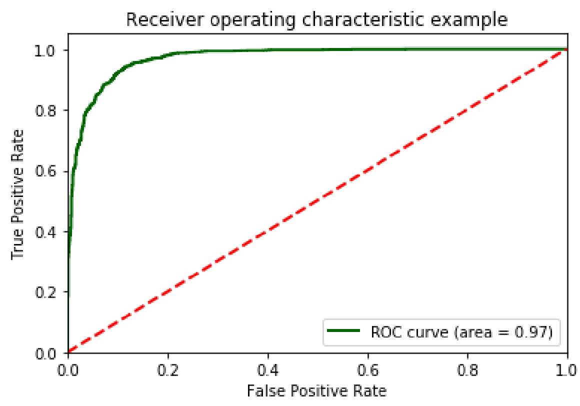
```
The accuracy score is: 0.6622061482820977
The recall score is: 0.8364779874213837
The precision score is: 0.6636726546906188
```

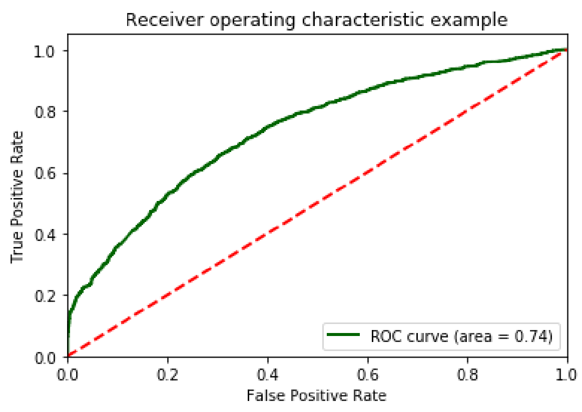
And then we set up regularization coefficients as 0.01, 0.1, 1, 10, 100, 1000, 10000 to get the confusion matrix and evaluation measures.

```
coef: 0.01
[[  95 1080]
 [   0 1590]]
The accuracy score is: 0.6094032549728752
The recall score is: 1.0
The precision score is: 0.5955056179775281

coef: 0.1
[[ 818  357]
 [  11 1579]]
The accuracy score is: 0.8669077757685353
The recall score is: 0.9930817610062893
The precision score is: 0.815599173553719

coef: 1
[[ 925  250]
 [  46 1544]]
The accuracy score is: 0.8929475587703436
The recall score is: 0.9710691823899371
The precision score is: 0.8606465997770345

coef: 10
[[ 945  230]
 [  96 1494]]
The accuracy score is: 0.8820976491862568
The recall score is: 0.939622641509434
The precision score is: 0.8665893271461717

coef: 100
[[ 931  244]
 [ 135 1455]]
The accuracy score is: 0.8629294755877034
The recall score is: 0.9150943396226415
The precision score is: 0.8563861094761624

coef: 1000
[[ 891  284]
 [ 159 1431]]
The accuracy score is: 0.8397830018083182
The recall score is: 0.9
The precision score is: 0.8344023323615161

coef: 10000
[[ 879  296]
 [ 170 1420]]
The accuracy score is: 0.8314647377938518
The recall score is: 0.8930817610062893
The precision score is: 0.8275058275058275
```

problem i:

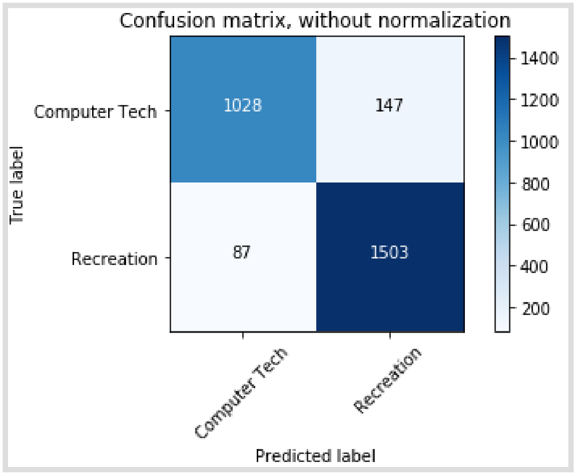
We change the norm regularization to l1 and perform the same operation as above.

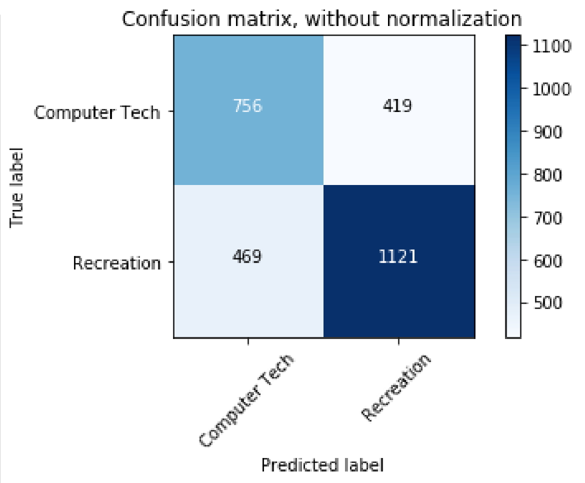




```
Confusion matrix, without normalization
[[1028  147]
 [  87 1503]]
Normalized confusion matrix
[[0.87 0.13]
 [0.05 0.95]]
```

```
Confusion matrix, without normalization
[[ 756  419]
 [ 469 1121]]
Normalized confusion matrix
[[0.64 0.36]
 [0.29 0.71]]
```





```
The accuracy score is: 0.6788426763110308
The recall score is: 0.7050314465408805
The precision score is: 0.727922077922078
```

```
The accuracy score is: 0.915370705244123
The recall score is: 0.9452830188679245
The precision score is: 0.9109090909090909
```

Then we change the regularization coefficients, we get the confusion matrix and evaluation data as following:

coef: 0.01
[[847 328]
 [21 1569]]
The accuracy score is: 0.8737793851717902
The recall score is: 0.9867924528301887
The precision score is: 0.8270954138112809

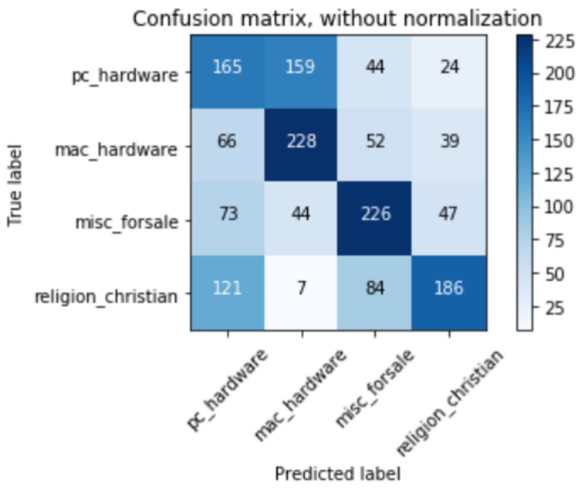
```
coef: 0.1
[[1026  149]
 [ 112 1478]]
The accuracy score is: 0.9056057866184448
The recall score is: 0.929559748427673
The precision score is: 0.9084204056545789

coef: 1
[[1028  147]
 [  87 1503]]
The accuracy score is: 0.915370705244123
The recall score is: 0.9452830188679245
The precision score is: 0.9109090909090909

coef: 10
[[ 952  223]
 [ 146 1444]]
The accuracy score is: 0.8665461121157324
The recall score is: 0.9081761006289308
The precision score is: 0.8662267546490702

coef: 100
[[ 883  292]
 [ 165 1425]]
The accuracy score is: 0.8347197106690778
The recall score is: 0.8962264150943396
The precision score is: 0.8299359347699475

coef: 1000
[[ 873  302]
 [ 172 1418]]
The accuracy score is: 0.8285714285714286
The recall score is: 0.8918238993710692
The precision score is: 0.8244186046511628

coef: 10000
[[ 872  303]
 [ 171 1419]]
The accuracy score is: 0.8285714285714286
The recall score is: 0.8924528301886793
The precision score is: 0.8240418118466899
```

As it shows above, for our data norm l1 has accuracy score. For the coefficients, too small or too large will not get a good result. People who want to get higher accuracy score will prefer norm l1 and people who want to get higher recall score will prefer norm l2.

Problem j: Multiclass Classification
Multiclass Naive Bayes

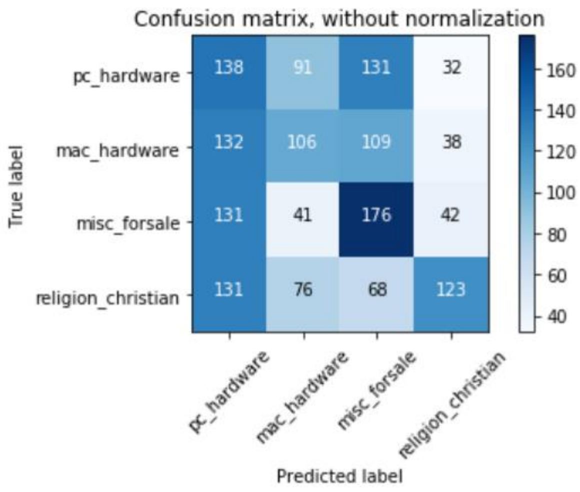
We implemented Gaussian Naive Bayes based on LSI and NMF to compare the difference. Figures on the left side is based on LSI while the right side is NMF.

```
Confusion matrix, without normalization
[[165 159  44  24]
 [ 66 228  52  39]
 [ 73  44 226  47]
 [121   7  84 186]]
Normalized confusion matrix
[[0.42 0.41 0.11 0.06]
 [0.17 0.59 0.14 0.1 ]
 [0.19 0.11 0.58 0.12]
 [0.3  0.02 0.21 0.47]]
```



```
The accuracy is 0.5143769968051118
The recall is 0.5143769968051118
The precision is 0.5238258050172109
```

```
Confusion matrix, without normalization
[[138  91 131  32]
 [132 106 109  38]
 [131  41 176  42]
 [131  76  68 123]]
Normalized confusion matrix
[[0.35 0.23 0.33 0.08]
 [0.34 0.28 0.28 0.1 ]
 [0.34 0.11 0.45 0.11]
 [0.33 0.19 0.17 0.31]]
```

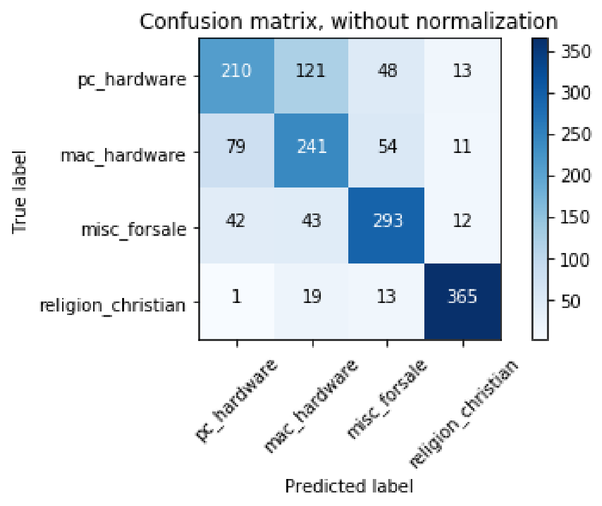


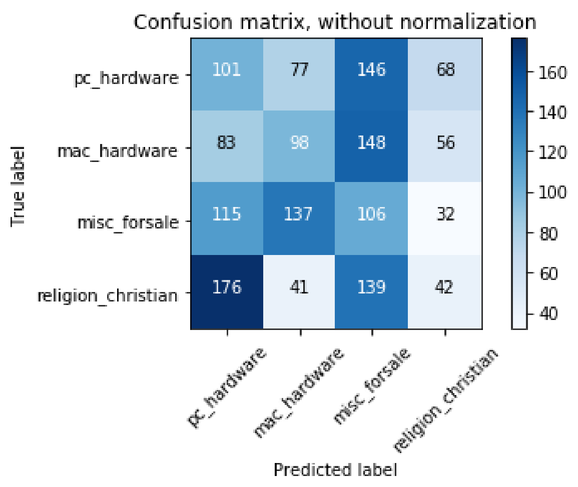
```
The accuracy is 0.3469648562300319
The recall is 0.3469648562300319
The precision is 0.371747884234991
```

SVM OVO:

```
Confusion matrix, without normalization
[[210 121  48  13]
 [ 79 241  54  11]
 [ 42  43 293  12]
 [  1  19  13 365]]
Normalized confusion matrix
[[0.54 0.31 0.12 0.03]
 [0.21 0.63 0.14 0.03]
 [0.11 0.11 0.75 0.03]
 [0.   0.05 0.03 0.92]]
```

```
Confusion matrix, without normalization
[[101  77 146  68]
 [ 83  98 148  56]
 [115 137 106  32]
 [176  41 139  42]]
Normalized confusion matrix
[[0.26 0.2  0.37 0.17]
 [0.22 0.25 0.38 0.15]
 [0.29 0.35 0.27 0.08]
 [0.44 0.1  0.35 0.11]]
```





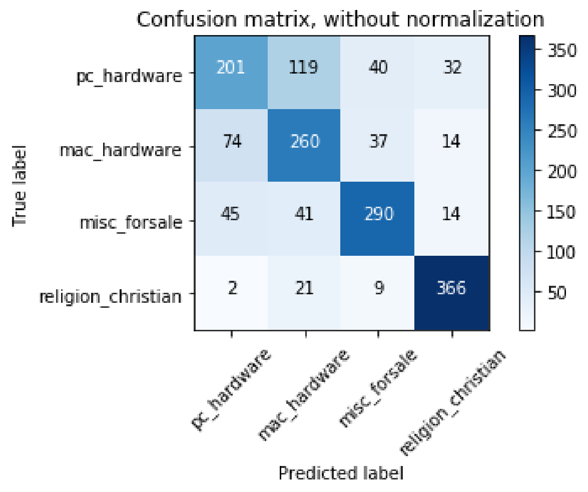
```
The accuracy is  0.708626198083067
The recall is  0.708626198083067
The precision is  0.7087074827436317
```

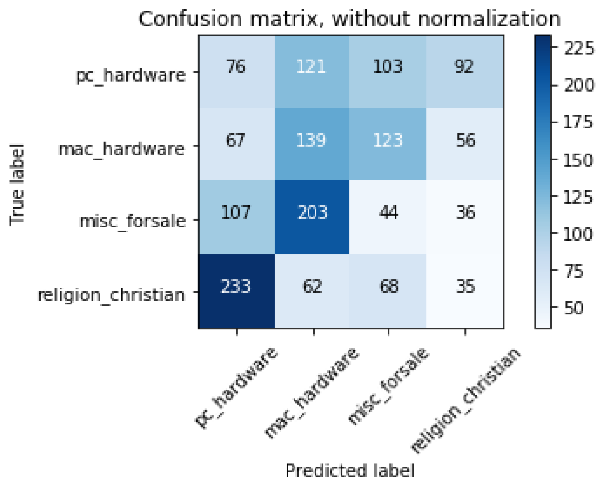
```
The accuracy is 0.22172523961661342
The recall is 0.2217252396166134
The precision is 0.2245094327086189
```

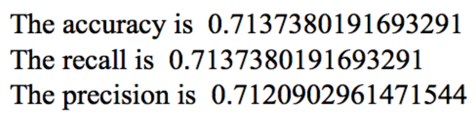
SVM OVR:

```
Confusion matrix, without normalization
[[201 119  40  32]
 [ 74 260  37  14]
 [ 45  41 290  14]
 [  2  21   9 366]]
Normalized confusion matrix
[[0.51 0.3  0.1  0.08]
 [0.19 0.68 0.1  0.04]
 [0.12 0.11 0.74 0.04]
 [0.01 0.05 0.02 0.92]]
```

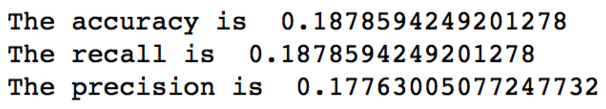
```
Confusion matrix, without normalization
[[ 76 121 103  92]
 [ 67 139 123  56]
 [107 203  44  36]
 [233  62  68  35]]
Normalized confusion matrix
[[0.19 0.31 0.26 0.23]
 [0.17 0.36 0.32 0.15]
 [0.27 0.52 0.11 0.09]
 [0.59 0.16 0.17 0.09]]
```







```
The accuracy is  0.7137380191693291
The recall is  0.7137380191693291
The precision is  0.7120902961471544
```

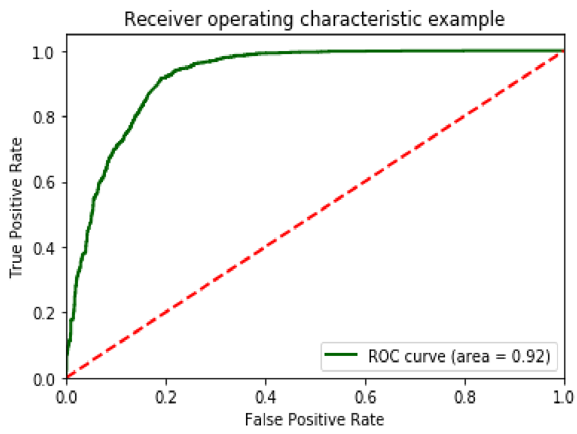


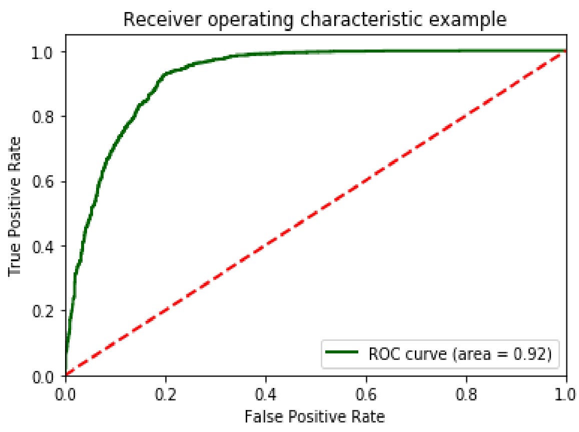
```
The accuracy is  0.1878594249201278
The recall is  0.1878594249201278
The precision is  0.17763005077247732
```

Conclusion 1: For LSI and NMF, as it shows above, LSI has better performance.

Now, we compare min_df=2 with min_df=5.The following figures are based on min_df=2 with LSI. Figures on the left side are min_df=2, the others are min_df=5.

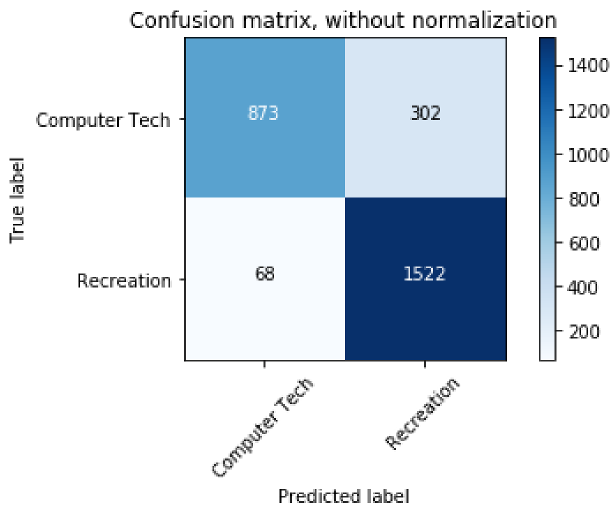
problem e:

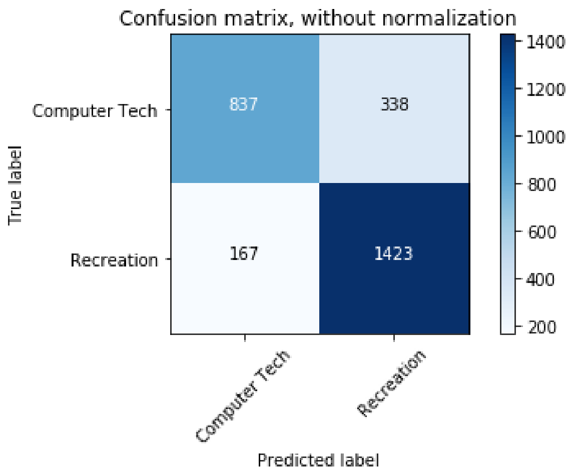




```
Confusion matrix, without normalization
[[ 873  302]
 [  68 1522]]
Normalized confusion matrix
[[0.74 0.26]
 [0.04 0.96]]
```

```
Confusion matrix, without normalization
[[ 837  338]
 [ 167 1423]]
Normalized confusion matrix
[[0.71 0.29]
 [0.11 0.89]]
```

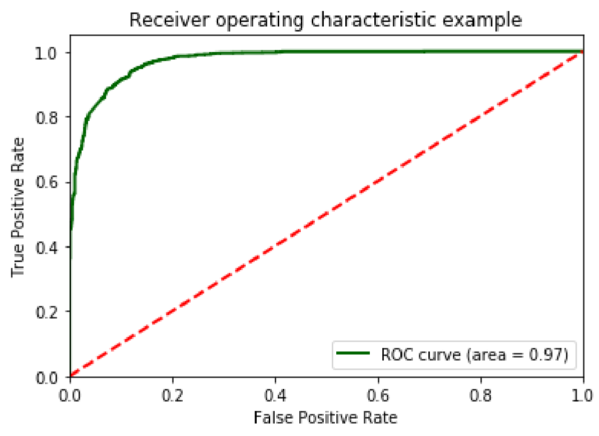


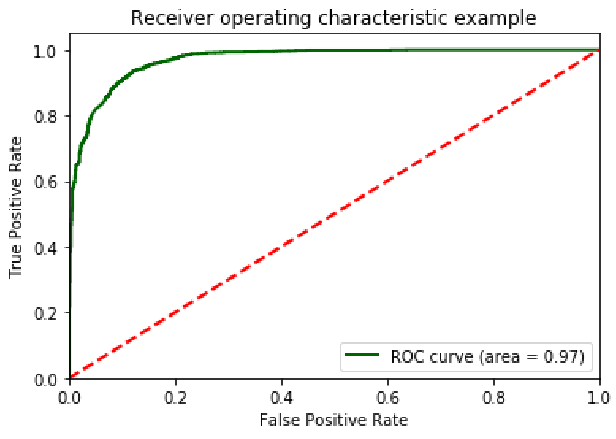


```
The accuracy score is: 0.8661844484629295
The recall score is: 0.9572327044025157
The precision score is: 0.8344298245614035
```

```
The accuracy score is: 0.8173598553345389
The recall score is: 0.8949685534591195
The precision score is: 0.8080636002271436
```

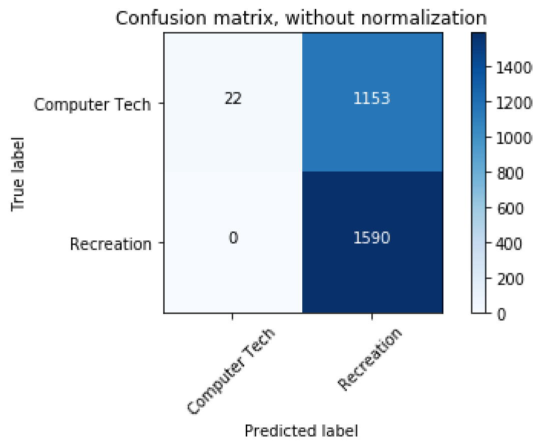
soft margin SVC:

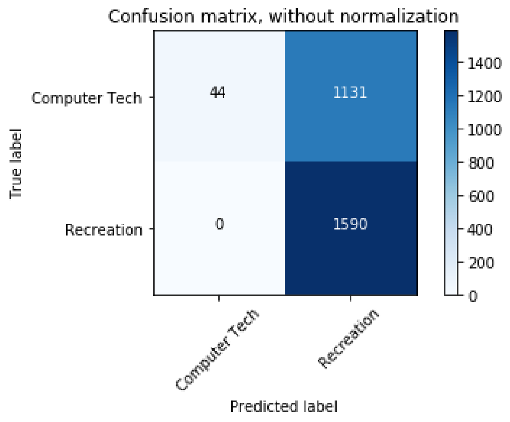




```
Confusion matrix, without normalization
[[  22 1153]
 [    0 1590]]
Normalized confusion matrix
[[0.02 0.98]
 [0.   1.  ]]
```

```
Confusion matrix, without normalization
[[  44 1131]
 [   0 1590]]
Normalized confusion matrix
[[0.04 0.96]
 [0.   1.  ]]
```





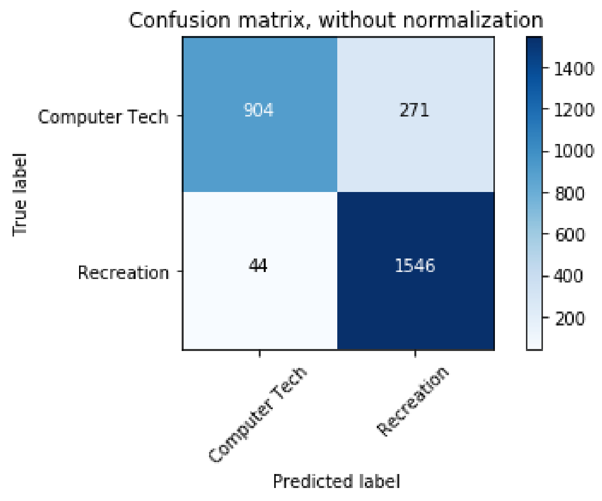
```
The accuracy score is: 0.583001808318264
The recall score is: 1.0
The precision score is: 0.5796573095151294
```

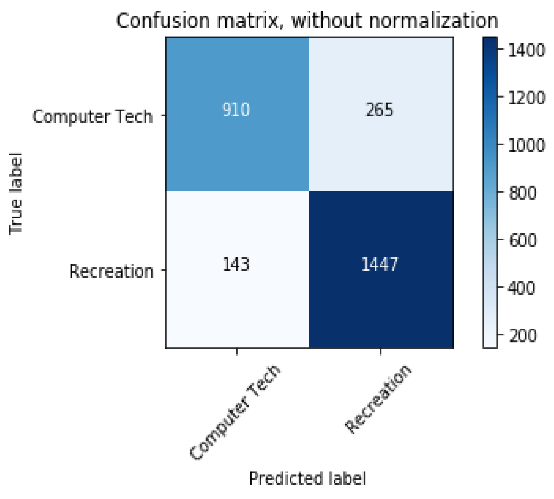
```
The accuracy score is: 0.5909584086799277
The recall score is: 1.0
The precision score is: 0.5843439911797134
```

problem f:

```
Confusion matrix, without normalization
[[ 904  271]
 [  44 1546]]
Normalized confusion matrix
[[0.77 0.23]
 [0.03 0.97]]
```

```
Confusion matrix, without normalization
[[ 910  265]
 [ 143 1447]]
Normalized confusion matrix
[[0.77 0.23]
 [0.09 0.91]]
```

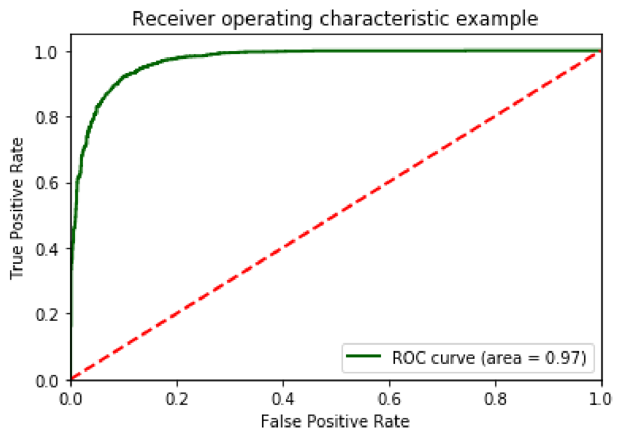


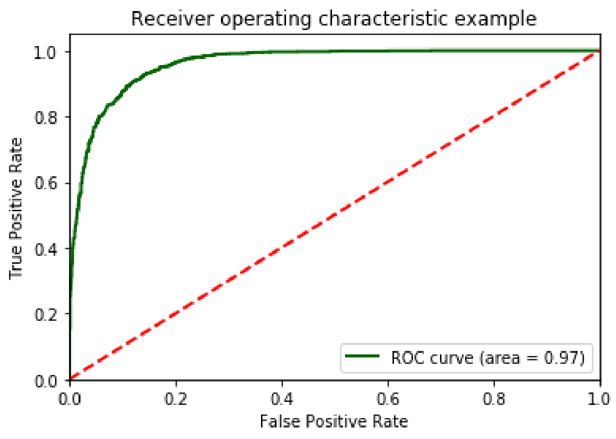


```
The accuracy score is: 0.583001808318264
The recall score is: 1.0
The precision score is: 0.5796573095151294
```

```
The accuracy score is: 0.5909584086799277
The recall score is: 1.0
The precision score is: 0.5843439911797134
```

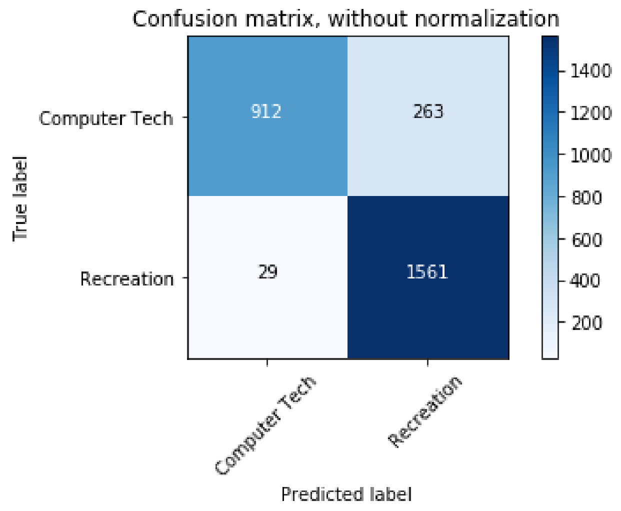
problem h:

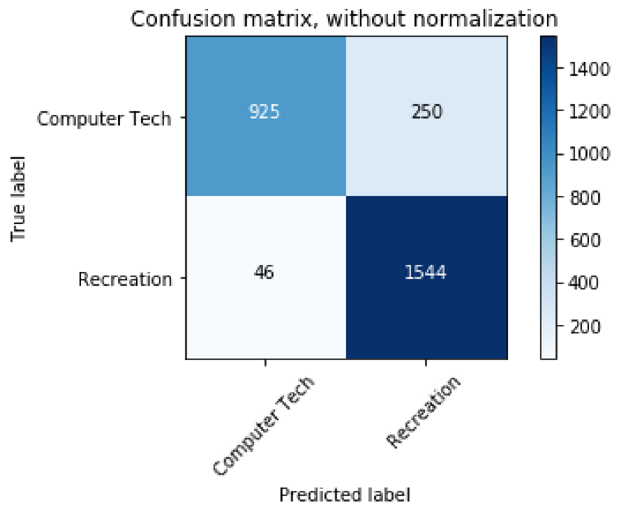




```
Confusion matrix, without normalization
[[ 912  263]
 [  29 1561]]
Normalized confusion matrix
[[0.78 0.22]
 [0.02 0.98]]
```

```
Confusion matrix, without normalization
[[ 925  250]
 [  46 1544]]
Normalized confusion matrix
[[0.79 0.21]
 [0.03 0.97]]
```

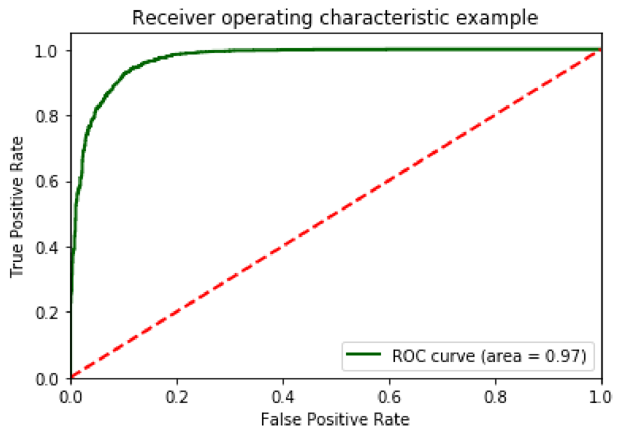


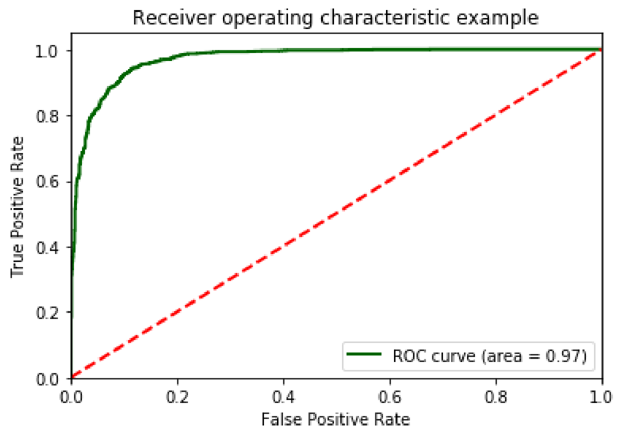


```
The accuracy score is: 0.8943942133815551
The recall score is: 0.9817610062893082
The precision score is: 0.8558114035087719
```

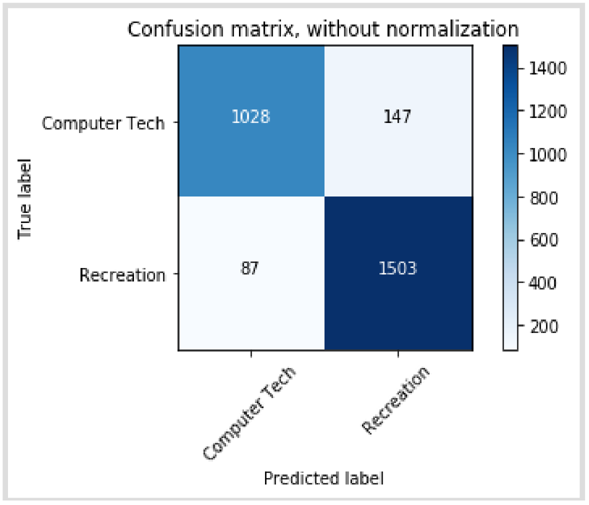
```
The accuracy score is: 0.8929475587703436
The recall score is: 0.9710691823899371
The precision score is: 0.8606465997770345
```

problem i:



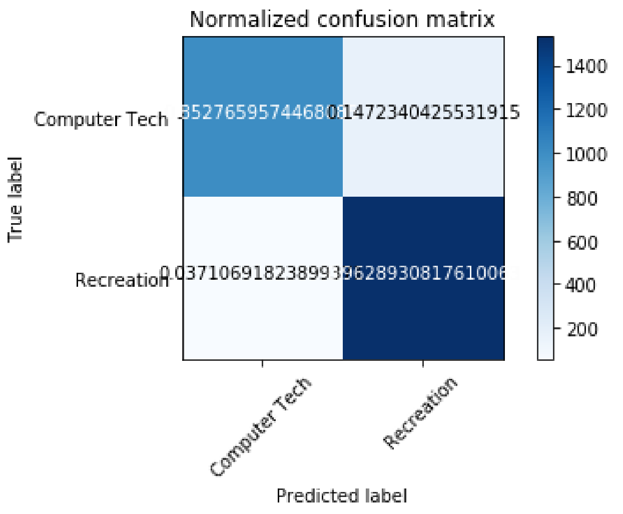


```
Confusion matrix, without normalization
[[1028  147]
 [  87 1503]]
Normalized confusion matrix
[[0.87 0.13]
 [0.05 0.95]]
```



```
The accuracy score is: 0.9160940325497288
The recall score is: 0.9628930817610063
The precision score is: 0.8984741784037559
```

```
Confusion matrix, without normalization
[[1002  173]
 [  59 1531]]
Normalized confusion matrix
[[0.85 0.15]
 [0.04 0.96]]
```



```
The accuracy score is: 0.915370705244123
The recall score is: 0.9452830188679245
The precision score is: 0.9109090909090909
```

Conclusion2 : From the figures above, we can conclude that min_df=2 has better performance than min_df=5. Keeping ‘rare’ terms is essential since some ‘rare’ term may convey important information and we should not miss them when analyze the text.